



## 【特許請求の範囲】

## 【請求項 1】

複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機において、  
前記各プロセッサが、  
自プロセッサで生成されたスレッドが、該スレッドが属するプロセス中の最初のスレッドである場合、前記主記憶装置上に、前記プロセスに固有の並列実行時間テーブルであって、複数のスレッド数毎に、そのスレッド数以上のスレッドに同時にプロセッサが割り当てられた時間を登録する領域を有する並列実行時間テーブルを生成する手段と、  
自プロセッサを前記プロセス内のスレッドに割り当てるとき、及び自プロセッサを前記プロセス内のスレッドから返還させるとき、現時点においてプロセッサが割り当てられている、前記プロセス内のスレッドの数と、現時点の時刻と、現時点における前記並列実行時間テーブルの内容と、前記並列実行時間テーブルに対する最新の更新時刻とに基づいて、前記並列実行時間テーブルを更新する手段と、  
前記並列実行時間テーブルの内容を取得する手段とを備えたことを特徴とするマルチプロセッサ計算機。

## 【請求項 2】

請求項 1 記載のマルチプロセッサ計算機において、  
前記並列実行時間テーブルが、前記主記憶装置以外の、前記複数のプロセッサによって共有される記憶装置上に生成されることを特徴とするマルチプロセッサ計算機。

## 【請求項 3】

複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機において、  
前記各プロセッサが、  
自プロセッサで生成されたスレッドが、該スレッドが属するプロセス中の最初のスレッドである場合、前記主記憶装置上に、前記プロセスに固有のスレッド共有空間であって、複数のスレッド数毎に、そのスレッド数以上のスレッドに同時にプロセッサが割り当てられていた時間が登録される領域を有する並列実行時間テーブルと、前記プロセス内のスレッドの内の、現時点においてプロセッサが割り当てられているスレッドの数が設定される並列カウンタと、前記並列実行時間テーブルに対する最新の更新時刻が設定されるタイムスタンプとを含むスレッド共有空間を生成する手段と、  
自プロセッサを前記プロセス内のスレッドに割り当てるとき、現在時刻と、前記並列実行時間テーブルの内容と、前記並列カウンタの内容と、前記タイムスタンプの内容とに基づいて、前記並列実行時間テーブルの内容を更新すると共に、前記タイムスタンプに現在時刻を設定し、更に、前記並列カウンタの値を増加させる手段と、  
自プロセッサを前記プロセス内のスレッドから返還させるとき、現在時刻と、前記並列実行時間テーブルの内容と、前記並列カウンタの内容と、前記タイムスタンプの内容とに基づいて、前記並列実行時間テーブルの内容を更新すると共に、前記タイムスタンプに現在時刻を設定し、更に、前記並列カウンタの値を減少させる手段と、  
前記並列実行時間テーブルの内容を取得する手段とを備えたことを特徴とするマルチプロセッサ計算機。

## 【請求項 4】

請求項 3 記載のマルチプロセッサ計算機において、  
前記スレッド共有空間が、前記主記憶装置以外の、前記複数のプロセッサによって共有される記憶装置上に生成されることを特徴とするマルチプロセッサ計算機。

## 【請求項 5】

複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機において、  
前記各プロセッサが、  
自プロセッサで生成されたプロセスが、該プロセスが属するジョブ中の最初のプロセスで

ある場合、前記主記憶装置上に、前記ジョブに固有の並列実行時間テーブルであって、複数のプロセス数毎に、そのプロセス数以上のプロセスに同時にプロセッサが割り当てられた時間を登録する領域を有する並列実行時間テーブルを生成する手段と、  
自プロセッサを前記ジョブ内のプロセスに割り当てるとき、及び自プロセッサを前記ジョブ内のプロセスから返還させるとき、現時点においてプロセッサが割り当てられている、前記ジョブ内のプロセスの数と、現時点の時刻と、現時点における前記並列実行時間テーブルの内容と、前記並列実行時間テーブルに対する最新の更新時刻とに基づいて、前記並列実行時間テーブルを更新する手段と、  
前記並列実行時間テーブルの内容を取得する手段とを備えたことを特徴とするマルチプロセッサ計算機。

10

**【請求項 6】**

複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機において、  
前記各プロセッサが、  
自プロセッサで生成されたプロセスが、該プロセスが属するジョブ中の最初のプロセスである場合、前記主記憶装置上に、前記ジョブに固有の共有空間であって、複数のプロセス数毎に、そのプロセス数以上のプロセスに同時にプロセッサが割り当てられていた時間が登録される領域を有する並列実行時間テーブルと、前記ジョブ内のプロセスの内の、現時点においてプロセッサが割り当てられているプロセスの数が設定される並列カウンタと、前記並列実行時間テーブルに対する最新の更新時刻が設定されるタイムスタンプとを含む  
プロセス共有空間を生成する手段と、  
自プロセッサを前記ジョブ内のプロセスに割り当てるとき、現在時刻と、前記並列実行時間テーブルの内容と、前記並列カウンタの内容と、前記タイムスタンプの内容とに基づいて、前記並列実行時間テーブルの内容を更新すると共に、前記タイムスタンプに現在時刻を設定し、更に、前記並列カウンタの値を増加させる手段と、  
自プロセッサを前記ジョブ内のプロセスから返還させるとき、現在時刻と、前記並列実行時間テーブルの内容と、前記並列カウンタの内容と、前記タイムスタンプの内容とに基づいて、前記並列実行時間テーブルの内容を更新すると共に、前記タイムスタンプに現在時刻を設定し、更に、前記並列カウンタの値を減少させる手段と、  
前記並列実行時間テーブルの内容を取得する手段とを備えたことを特徴とするマルチプロセッサ計算機。

20

30

**【請求項 7】**

複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機の構成要素であるプロセッサを、  
自プロセッサで生成されたスレッドが、該スレッドが属するプロセス中の最初のスレッドである場合、前記主記憶装置上に、前記プロセスに固有の並列実行時間テーブルであって、複数のスレッド数毎に、そのスレッド数以上のスレッドに同時にプロセッサが割り当てられた時間を登録する領域を有する並列実行時間テーブルを生成する手段、  
自プロセッサを前記プロセス内のスレッドに割り当てるとき、及び自プロセッサを前記プロセス内のスレッドから返還させるとき、現時点においてプロセッサが割り当てられている、前記プロセス内のスレッドの数と、現時点の時刻と、現時点における前記並列実行時間テーブルの内容と、前記並列実行時間テーブルに対する最新の更新時刻とに基づいて、前記並列実行時間テーブルを更新する手段、  
前記並列実行時間テーブルの内容を取得する手段として機能させるためのプログラム。

40

**【請求項 8】**

複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機の構成要素であるプロセッサを、  
自プロセッサで生成されたプロセスが、該プロセスが属するジョブ中の最初のプロセスである場合、前記主記憶装置上に、前記ジョブに固有の並列実行時間テーブルであって、複数のプロセス数毎に、そのプロセス数以上のプロセスに同時にプロセッサが割り当てられ

50

た時間を登録する領域を有する並列実行時間テーブルを生成する手段、  
自プロセッサを前記ジョブ内のプロセスに割り当てるとき、及び自プロセッサを前記ジョブ内のプロセスから返還させるとき、現時点においてプロセッサが割り当てられている、前記ジョブ内のプロセスの数と、現時点の時刻と、現時点における前記並列実行時間テーブルの内容と、前記並列実行時間テーブルに対する最新の更新時刻とに基づいて、前記並列実行時間テーブルを更新する手段、  
前記並列実行時間テーブルの内容を取得する手段として機能させるためのプログラム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

10

本発明は、複数のプロセッサが主記憶装置を共有するマルチプロセッサ計算機に関し、特に、並列プログラムが如何に効率良く動作したかを表す指標である並列実行時間を算出する技術に関する。

【0002】

【従来の技術】

複数のプロセッサから構成されるマルチプロセッサ計算機の性能評価を行う従来の技術として、各プロセッサ毎にプロセッサが稼働状態にあるときの開始時刻及び終了時刻と、プロセッサが非稼働状態のときの時刻とを記録し、制御部が、上記各時刻に基づいて任意の時刻におけるプロセッサの稼働台数や、稼働率を求める技術が知られている（例えば、特許文献1参照）。

20

【0003】

【特許文献1】

特開平5-189395号公報

【0004】

【発明が解決しようとする課題】

上述した従来の技術によれば、何台のプロセッサが同時に稼働しているかを知ることができないが、マルチプロセッサ計算機を使用して並列プログラムを実行するユーザにとって重要なことは、プロセッサが何台同時に稼働しているかではなく、並列プログラムが効率的に実行されるか否かである。マルチプロセッサ計算機で実行される並列プログラムが1つの場合には、上述した従来の技術によっても並列プログラムが効率的に実行されているか否かを判定することができる。つまり、多くのプロセッサが同時に稼働している時間が長いほど、並列プログラムが効率的に実行されていると判定することができる。

30

【0005】

しかし、マルチプロセッサ計算機で実行される並列プログラムが複数の場合は、プロセッサの稼働台数が分かったとしても、稼働台数に基づいて並列プログラムが効率的に実行されているか否かを判定することはできない。例えば、並列プログラム $\alpha$ の2つのスレッドと、並列プログラム $\beta$ の1つのスレッドとがマルチプロセッサ計算機で実行されている場合、上述した従来の技術では、プロセッサの稼働台数が3台であることが分かるだけであり、並列プログラム $\alpha$ 、 $\beta$ 毎に、同時に実行されているスレッド数が分からないため、各並列プログラムが効率的に実行されているか否かを判定することはできない。

40

【0006】

そこで、本発明の目的は、複数の並列プログラムが同時に実行されている場合でも、各並列プログラムが効率的に実行されているか否かを判定できるようにすることにある。

【0007】

【課題を解決するための手段】

本発明にかかる第1のマルチプロセッサ計算機は、上記目的を達成するため、複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機において、  
前記各プロセッサが、  
自プロセッサで生成されたスレッドが、該スレッドが属するプロセス中の最初のスレッド

50

である場合、前記主記憶装置上に、前記プロセスに固有の並列実行時間テーブルであって、複数のスレッド数毎に、そのスレッド数以上のスレッドに同時にプロセッサが割り当てられた時間を登録する領域を有する並列実行時間テーブルを生成する手段と、  
自プロセッサを前記プロセス内のスレッドに割り当てるとき、及び自プロセッサを前記プロセス内のスレッドから返還させるとき、現時点においてプロセッサが割り当てられている、前記プロセス内のスレッドの数と、現時点の時刻と、現時点における前記並列実行時間テーブルの内容と、前記並列実行時間テーブルに対する最新の更新時刻とに基づいて、前記並列実行時間テーブルを更新する手段と、  
前記並列実行時間テーブルの内容を取得する手段とを備えたことを特徴とする。

【0008】

10

また、本発明にかかる第2のマルチプロセッサ計算機は、  
第1のマルチプロセッサ計算機において、  
前記並列実行時間テーブルが、前記主記憶装置以外の、前記複数のプロセッサによって共有される記憶装置上に生成されることを特徴とする。

【0009】

より具体的には、本発明にかかる第3のマルチプロセッサ計算機は、  
複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機において、  
前記各プロセッサが、

自プロセッサで生成されたスレッドが、該スレッドが属するプロセス中の最初のスレッドである場合、前記主記憶装置上に、前記プロセスに固有のスレッド共有空間であって、複数のスレッド数毎に、そのスレッド数以上のスレッドに同時にプロセッサが割り当てられていた時間が登録される領域を有する並列実行時間テーブルと、前記プロセス内のスレッドの内の、現時点においてプロセッサが割り当てられているスレッドの数が設定される並列カウンタと、前記並列実行時間テーブルに対する最新の更新時刻が設定されるタイムスタンプとを含むスレッド共有空間を生成する手段と、

20

自プロセッサを前記プロセス内のスレッドに割り当てるとき、現在時刻と、前記並列実行時間テーブルの内容と、前記並列カウンタの内容と、前記タイムスタンプの内容とに基づいて、前記並列実行時間テーブルの内容を更新すると共に、前記タイムスタンプに現在時刻を設定し、更に、前記並列カウンタの値を増加させる手段と、

30

自プロセッサを前記プロセス内のスレッドから返還させるとき、現在時刻と、前記並列実行時間テーブルの内容と、前記並列カウンタの内容と、前記タイムスタンプの内容とに基づいて、前記並列実行時間テーブルの内容を更新すると共に、前記タイムスタンプに現在時刻を設定し、更に、前記並列カウンタの値を減少させる手段と、  
前記並列実行時間テーブルの内容を取得する手段とを備えたことを特徴とする。

【0010】

本発明にかかる第4のマルチプロセッサ計算機は、  
第3のマルチプロセッサ計算機において、  
前記スレッド共有空間が、前記主記憶装置以外の、前記複数のプロセッサによって共有される記憶装置上に生成されることを特徴とする。

40

【0011】

【作用】

或るプロセッサでスレッドが生成されると、そのスレッドがプロセス中の最初のスレッドである場合、主記憶装置上に上記プロセス固有の並列実行時間テーブルが作成される。その後、何れかのプロセッサが、上記プロセス内のスレッドにプロセッサを割り当てたり、上記プロセス内のスレッドからプロセッサを返還させるとき、現時点の時刻と、上記並列実行時間テーブルに対する最新の更新時刻と、現時点においてプロセッサが割り当てられている上記プロセスに属するスレッドの数と、並列実行時間テーブルの内容とに基づいて並列実行時間テーブルを更新する。このように、プロセス固有の並列実行時間テーブルを用いて並列実行時間を管理するようにしているので、マルチプロセッサ計算機で複数の並

50

列プログラムが同時に実行されている場合であっても、各並列プログラムが効率的に動作しているか否かを判定することが可能になる。

【0012】

【発明の実施の形態】

次に本発明の実施の形態について図面を参照して詳細に説明する。

【0013】

【実施例の構成】

図1は本発明にかかるマルチプロセッサ計算機の実施例のブロック図であり、複数のプロセッサ（CPU、演算器）100-1～100-Nと、各プロセッサ100-1～100-Nによって共有される主記憶装置200とから構成されている。

10

【0014】

先ず、以下の説明で使用する用語の定義について述べておく。

【0015】

プロセスとは、UNIX（登録商標）システムにおいて一般的にプロセスと呼ばれる概念に相当する。すなわち、プロセスとは、固有の記憶領域を有する最小の実行単位であり、時分割処理によりスケジューリングされるものである。従って、マルチプロセッサ計算機においては、複数のプロセッサ上で同時に異なるプロセスを実行することができ、シングルプロセッサシステムにおいても複数のプロセスをインタラクティブに実行することができる。

【0016】

また、スレッドとは、UNIX（登録商標）システムにおいて一般的にスレッドと呼ばれる概念に相当する。すなわち、スレッドは単一のプロセスから生成された複数の実行単位であり、プロセスと同様に時分割処理によりスケジューリングされる。スレッドは固有の記憶領域を持たない最小の実行単位であり、同一プロセスに属する各スレッドはそれらが属している同一プロセスに固有の記憶領域を共有している。

20

【0017】

また、時分割処理とは、非同期的あるいは同期的に発生する割り込みを契機として、実行待ち状態のプロセスのうち実行優先度の高いものからプロセッサが割り当てられ、プロセッサ上で実行される間に実行優先度が低下し、優先度が低いものからプロセッサを返還して実行待ち状態に移行するスケジューリング方式のことである。一般的にはマルチプロセッサ計算機における並列プログラムの最小実行単位はスレッドにより実現される事が多いが、プロセスによって実現される場合もある。スレッドにより実現される場合は、並列プログラムは複数のスレッドにより構成されたプロセスに相当し、このプロセスに固有の空間でかつこのプロセスを構成する複数のスレッドが互いに共有するスレッド共有空間を利用して動作する。プロセスにより実現される場合は、並列プログラムは複数のプロセスにより構成されたジョブに相当し、このジョブに固有の空間でかつこのジョブを構成する複数のプロセスが互いに共有するプロセス共有空間を利用して動作する。以下では並列プログラムの最小の実行単位がスレッドである事を想定した記述となっているが、最小の実行単位がプロセスであるようなシステムの場合でも、プロセスをスレッド、ジョブをプロセスと読み替えることで同様の効果を得ることができる。

30

40

【0018】

図1において、主記憶装置200は、各プロセッサ100-1～100-Nから等価に共有されたシステムに唯一の主記憶装置である。主記憶装置200は、スレッド共有空間201と、システムタイマ205とを含んでいる。

【0019】

スレッド共有空間201は、任意のある時点においてシステムに存在する複数のプロセスの内の任意の一つが占有してアクセス可能な記憶領域であり、なおかつそのプロセスに属する各スレッドが共有している記憶領域である。このようなスレッド共有空間201には、並列実行時間テーブル202と、並列カウンタ203と、タイムスタンプ204とが設けられている。

50

## 【0020】

並列実行時間テーブル202は、対応するプロセスを構成しているスレッドに関する1並列実行時間からN並列実行時間までの配列（要素数N）が格納される領域を表している。ここで、n並列実行時間（ $n = 1, 2, \dots, N$ ）とはn個以上のスレッドが同時に動作していた時間（n個以上のスレッドに同時にプロセッサが割り当てられていた時間）を表すものとする。

## 【0021】

並列カウンタ203は、対応するプロセスを構成するスレッドの内、同時に異なるプロセッサで動作しているスレッドの数（0～N）を表している。

## 【0022】

タイムスタンプ204には、並列実行時間テーブル202に対する最新の更新時刻が格納される。

## 【0023】

システムタイマ205には、システム稼働中にマシクロック単位で単調増加し続けるシステムに唯一のタイマ値が格納される。

## 【0024】

また、図1においてプロセッサ100-1は、ユーザプログラム実行処理部101と、割込み開始処理部102と、並列実行時間算出処理部103と、並列カウンタ減少処理部104と、並列実行時間取得処理部105と、処理部106と、並列実行時間算出処理部107と、並列カウンタ増加処理部108と、割込み終了処理部109と、スレッド開始処理部110と、プロセッサ割当処理部111と、ウェイト処理部112と、プロセッサ返還処理部113と、スレッド終了処理部114と、記録媒体Kとを備えている。なお、他のプロセッサもプロセッサ100-1と同様の構成を有している。

## 【0025】

ユーザプログラム実行処理部101は、スレッドがユーザモードでプログラムのコードを実行する手段である。

## 【0026】

割込み開始処理部102は、システムコールやI/O処理、例外処理などの非同期的な割込みの発生に伴い、プロセッサコンテキストのセーブや割込み原因に対応する割込みハンドラを呼び出す。

## 【0027】

並列実行時間算出部103は、割込み開始処理の直後に呼び出され、システムコールの発行元のスレッドが属するプロセスに固有のスレッド共有空間201内の並列カウンタ203の値を参照し、その値が、「1」以上であった場合は、並列実行時間テーブル202内の第1番目のエントリから並列カウンタ203の値に対応するエントリまでの各値に、システムタイマ205とタイムスタンプ204との差分を加算し、その後、タイムスタンプ204にシステムタイマ205の値を設定する。また、並列カウンタ203の値が「0」であった場合は、タイムスタンプ204にシステムタイマ205の値を設定する処理のみを行い、並列実行時間テーブル202に対する更新処理は行わない。

## 【0028】

並列カウンタ減少処理部104は、並列カウンタ203の値を1減少させる機能を有する。

## 【0029】

並列実行時間取得処理部105は、並列実行時間取得システムコールにより呼び出され、発行元のスレッドが属するプロセスに対応するスレッド共有空間201の並列実行時間テーブル202の各エントリの内容を取得し、スレッドに返却する機能を有する。

## 【0030】

並列実行時間算出処理部107は、並列実行時間算出処理部103と同様の機能を有する。

## 【0031】

10

20

30

40

50

並列カウンタ増加処理部 108 は、並列カウンタ 203 の値を 1 増加させる機能を有する。

【0032】

割込み終了処理部 109 は、プロセッサコンテキストのリストアや割込み発生ポイントへの復帰を行う。

【0033】

スレッド開始処理部 110 は、新たに生成されたスレッドが、そのスレッドの属しているプロセス内の最初のスレッドであった場合、プロセスに固有の空間で且つそのプロセスに属している全てのスレッドが共有するスレッド共有空間 201 を確保した上で、並列実行時間テーブル 202、並列カウンタ 203 及びタイムスタンプ 204 の値を「0」に初期化する。

10

【0034】

プロセッサ割当処理部 111 は、他の実行優先度の低いスレッドからプロセッサを奪い取り、仮想空間の切り替えなどのスイッチ処理を実行してスレッドの実行を開始させる。

【0035】

ウェイト処理部 112 は、スリープシステムコールやタイムスライス切れが発生した場合、スレッドを実行待ちキューやスリープキューなどを用いて停止させる。

【0036】

プロセッサ返還処理部 113 は、システムコールや I/O 待ちによるスリープ呼び出しや、終了処理、タイムスライス切れ割込みで自スレッドよりも実行優先度の高いスレッドがある時などに呼び出され、他の実行優先度の高いスレッドにプロセッサを譲るために仮想空間の切り替えなどのスイッチ処理を実行する。

20

【0037】

スレッド終了処理部 114 は、終了したスレッドがプロセス内の最後のスレッドである場合は、上記プロセス固有のスレッド共有空間 201 を解放する機能を有する。

【0038】

なお、並列実行時間算出処理部 103、並列カウンタ減少処理部 104 によって連続的に行われる処理と、並列実行時間算出処理部 107、並列カウンタ増加処理部 108 によって連続的に行われる処理と、並列実行時間取得処理部 105 によって行われる処理とはそれぞれ互いに、同一プロセス内の異なるスレッドにおいて同時に実行されないよう、排他制御によりシリアライズされている。

30

【0039】

処理部 106 は、プロセッサ返還処理、並列実行時間取得処理以外の、システムコールに応じた処理、I/O 処理、例外処理などの一般的な割込み処理を行う。

【0040】

記録媒体 K は、ディスク、半導体メモリ、その他の記録媒体である。この記録媒体 K に記録されているプログラムは、プロセッサ 100-1 によって読み取られ、その動作を制御することで、プロセッサ 100-1 上に、ユーザプログラム実行処理部 101、割込み開始処理部 102、並列実行時間算出処理部 103、並列カウンタ減少処理部 104、並列実行時間取得処理部 105、処理部 106、並列実行時間算出処理部 107、並列カウンタ増加処理部 108、割込み終了処理部 109、スレッド開始処理部 110、プロセッサ割当処理部 111、ウェイト処理部 112、プロセッサ返還処理部 113、スレッド終了処理部 114 を実現する。

40

【0041】

【実施例の動作】

次に各図を参照して本実施例の動作について詳細に説明する。

【0042】

先ず、プロセッサ 100-1 においてスレッド X が生成されると、プロセッサ 100-1 内のスレッド開始処理部 110 は、スレッド X を実行待ちキュー（図示せず）につなぎ、更に、図 2 の流れ図に示すように、スレッド X が、それが属するプロセス Z 内の最初のス

50



レッドであるか否かを判断し（ステップ S 1）、最初のスレッドであった場合（ステップ S 1 が Y e s）は、上記プロセス Z に固有のスレッド共有空間 2 0 1 を生成し、更に、並列実行時間テーブル 2 0 2 の各エントリ、並列カウンタ 2 0 3 及びタイムスタンプ 2 0 4 の値を「0」に初期設定する（ステップ S 2）。これに対して、最初のスレッドでなかった場合（ステップ S 1 が N o）は、ステップ S 2 をスキップする。

【0043】

その後、或るプロセッサ（例えば、プロセッサ 1 0 0 - 1）内のプロセッサ割当処理部 1 1 1 によって、実行待ちキューにつながれているスレッド X にプロセッサ 1 0 0 - 1 が割り当てられると、並列実行時間算出処理部 1 0 7 が、並列実行時間の算出処理を行う（ステップ S 1 3）。

10

【0044】

このステップ S 1 3 の処理を詳しく説明すると、次のようになる。先ず、スレッド X が属するプロセス Z に割り当てられているスレッド共有空間 2 0 1 中の並列カウンタ 2 0 3 の値 C を参照し、その値 C が「1」以上であるか否かを調べる。

【0045】

そして、「1」以上であった場合は、システムタイマ 2 0 5 の値 S T M とタイムスタンプ 2 0 4 の値 T S との差分（ $S T M - T S$ ）を求める。その後、並列実行時間テーブル 2 0 2 の各エントリの内の、第 1 番目のエントリから並列カウンタ 2 0 3 の値に対応するエントリまでを対象にして、そこに設定されている値に上記差分（ $S T M - T S$ ）を加算する。そして、最後に、タイムスタンプ 2 0 4 の値 T S にシステムタイマ 2 0 5 の値 S T M を

20

【0046】

これに対して、並列カウンタ 2 0 3 の値 C が「1」未満であった場合は、タイムスタンプ 2 0 4 の値 T S にシステムタイマ 2 0 5 の値 S T M を設定する処理のみを行い、並列実行時間テーブル 2 0 2 に対する更新処理は行わない。以上がステップ S 1 3 で行う処理の詳細である。

【0047】

並列実行時間算出処理部 1 0 7 の処理が終了すると、並列カウンタ増加処理部 1 0 8 が、スレッド X が属しているプロセス Z に割り当てられているスレッド共有空間 2 0 1 内の並列カウンタ 2 0 3 の値 C を増加させる（ステップ S 1 4）。なお、本実施例では、並列カウンタ 2 0 3 の値 C を「1」増加させるものとする。

30

【0048】

その後、割込み終了処理部 1 0 9 による割込み終了処理が行われ、上記スレッド X は、ユーザモードに移行し、ユーザプログラム実行処理部 1 0 1 でユーザプログラムを実行する（ステップ S 3）。このステップ S 3 では、並列プログラムがプログラムの記述に従って実行され、割込みが発生しない限り（ステップ S 4 が Y e s とならない限り）、ユーザプログラムが実行され続ける。

【0049】

スレッド X がシステムコールや I / O を発行したり、タイムスライス切れや例外処理により割り込まれた場合（ステップ S 4 が Y e s）は、割込み開始処理部 1 0 2 に処理が移行し、以下の処理が行われる。

40

【0050】

先ず、並列実行時間算出処理部 1 0 3 がスレッド X が属しているプロセス Z のスレッド共有空間 2 0 1 を対象にして、ステップ S 1 3 において並列実行時間算出処理部 1 0 7 が行った処理と同様の処理を行う（ステップ S 7）。次いで、並列カウンタ減少処理部 1 0 4 が並列カウンタ 2 0 3 の値 C を減少させる（ステップ S 8）。本実施例では、並列カウンタ 2 0 3 の値 C を「1」減少させるものとする。

【0051】

その後、割込み要因に対応する割込み処理が実行される。もし、割込み要因が並列実行時間取得システムコールである場合（ステップ S 9 が Y e s）は、並列実行時間取得処理部

50

105において、並列実行時間の取得処理が行われる（ステップS11）。このステップS11の並列実行時間の取得処理では、並列実行時間テーブル202の内容を全て取得し、システムコールの発行元のスレッドに返却する処理が行われる。その後、並列実行時間算出処理部107において、スレッドXが属するプロセスZのスレッド共有空間201を処理対象にして前述した並列実行時間の算出処理が行われ（ステップS13）、更に、並列カウンタ増加処理部108において前述した並列カウンタ203の増加処理が行われる（ステップS14）。その後、割込み終了処理部109において割込み終了処理が実行され、上記スレッドXは、ユーザモードに移行し、ユーザプログラム実行処理部101でユーザプログラムを実行する（ステップS3）。

#### 【0052】

10

また、割込み要因が並列実行時間取得システムコールでなかった場合（ステップS9がNo）は、割込み要因が終了システムコールであるか否かを判定する（ステップS10）。

#### 【0053】

そして、終了システムコールでなかった場合（ステップS10がNo）は、処理部106において割込み要因に応じた割込み処理が行われる（ステップS12）。このステップS12で行われる処理としては、例えば、新たなスレッドYの生成処理などがある。もし、ステップS12で新たなスレッドYが生成されたとすると、スレッド開始処理部110は、スレッドYを実行待ちキューにつなぐと共に、上記スレッドYを処理対象にして前述したステップS1の処理を行う。この場合、スレッドYはプロセスZ中の最初のスレッドでない（ステップS1がNo）、ステップS2はスキップされる。その後、プロセッサ100-1～100-Nの内の、或る1台のプロセッサ100-n（ $1 \leq n \leq N$ ）内のプロセッサ割当処理部111によってスレッドYにプロセッサ100-nが割り当てられると、並列実行時間算出処理部107、並列カウンタ増加処理部108、割込み終了処理部109による処理が順次行われる。

20

#### 【0054】

これに対して、割込み要因が終了システムコールであった場合（ステップS10がYes）は、プロセッサ返還処理部113によるプロセッサ返還処理が行われた後、スレッド終了処理部114によるスレッド終了処理が行われる。このスレッド終了処理においては、先ず、終了システムコールの発行元のスレッド（例えば、スレッドX）が、同一プロセスZ内の最後のスレッドであるか否かを判断する（ステップS5）。そして、最後のスレッドである場合（ステップS5がYes）は、スレッドXが属しているプロセスZに割り当てられているスレッド共有空間201を解放する（ステップS6）。これに対して、最後のスレッドでない場合（ステップS5がNo）は、ステップS6をスキップする。その後、スレッドXは終了する。

30

#### 【0055】

以上の処理により、割込みの発生毎に並列実行時間テーブル202に1並列実行時間からN並列実行時間までの各値が適時採取されことになる。なお、並列実行時間を $T_n$ 、並列カウンタ203の値をC、タイムスタンプ204の値をTS、システムタイマ205の値をSTMと表記すると、ステップS7、S13の並列実行時間の算出処理は、 $T_n = T_n + (STM - TS)$ （但し $1 \leq n \leq C$ の各値について）、および $TS = STM$ と表記できる。また、ステップS8の並列カウンタ減算処理は $C = C - 1$ と表記でき、ステップS14の並列カウンタ増加処理は $C = C + 1$ と表記できる。

40

#### 【0056】

次に、具体例を用いて並列実行時間が算出される様子を説明する。

#### 【0057】

図3は、3つのスレッドA、B、Cにより構成される並列プログラムが動作した際の、並列実行時間テーブル202の各エントリの値 $T_n$ （ $n = 1, 2, \dots, N$ ）、並列カウンタ203の値C及びタイムスタンプ204の値TSの遷移を表した図である。なお、以下では説明を容易にするために、割り込みは時刻P1, P2, ..., P8の8箇所においてのみ発生したものとし、なおかつ割り込みが発生してから終了するまでに要する時間は十分に

50

短く0に等しかつたと仮定する。すなわち、割込み開始処理部102で割込み開始処理が行われてから割込み終了処理部109で割込み終了処理が行われるまでの間にシステムタイマ205の値は変動しなかったと仮定する。なお、現実のシステムにおいては、割り込みは任意のスレッドの任意の時点で発生し、割り込みが発生してから終了するまでに要する時間は0以上の可変値となるが、現実のシステムのどのような場合も図3で例示した処理の組み合わせにすぎず、並列実行時間が正しく採取される。

#### 【0058】

時刻P1 (STM=10)において、プロセッサ100-1でスレッドAが生成されると、プロセッサ100-1内のスレッド開始処理部110は、スレッドAを実行待ちキューにつなぐと共に、スレッドAが、それが属するプロセスD内の最初のスレッドであるか否かを判断する(図2、ステップS1)。この場合、スレッドAは最初のスレッドであるので(ステップS1がYes)、スレッド開始処理部110は、スレッドAが属するプロセスD固有のスレッド共有空間201を生成し、更に、並列実行時間テーブル202の各エントリの値T1~TN、並列カウンタ203の値C及びタイムスタンプ204の値TSを全て「0」に初期化する(ステップS2)。

10

#### 【0059】

その後、或るプロセッサ(例えば、プロセッサ100-1)内のプロセッサ割当処理部111によって、実行待ちキューにつながれているスレッドAにプロセッサ100-1が割り当てられると、並列実行時間算出処理部107が、ステップS13の並列実行時間算出処理を行う。この場合、並列カウンタ203の値Cは「0」であるので、並列実行時間算出処理部107は、並列実行時間テーブル202に対する更新処理は行わず、タイムスタンプ204の値TSにシステムタイマ205の値STMの値を設定する処理だけを行う。従って、TS=10となる。

20

#### 【0060】

その後、並列カウンタ増加処理部108が、並列カウンタ203の値Cを+1し、C=1とする(ステップS14)。その後、割込み終了処理部109による割込み終了処理が行われ、スレッドAがユーザプログラム実行処理部101でユーザプログラムの実行を開始する(ステップS3)。

#### 【0061】

次に、時刻P2 (STM=25)において、スレッドAが新たなスレッドBを生成するためにスレッド生成システムコールを発行する(ステップS4がYes)。これにより、割込み開始処理部102に処理が移行し、並列実行時間算出処理部103において並列実行時間算出処理が行われる(ステップS7)。このとき、並列カウンタ203の値Cは「1」であるので、並列実行時間テーブル202の第1番目のエントリの値T1に、システムタイマ205の値STMとタイムスタンプ204の値TSとの差分(STM-TS)を加算し、更に、システムタイマ205の値STMをタイムスタンプ204の値TSに設定する。すなわち、 $T1 = T1 + (STM - TS) = 0 + (25 - 10) = 15$ 、 $TS = STM = 25$ となる。その後、並列カウンタ減少処理部104が、並列カウンタ203の値Cを1減少させ、C=0とする(ステップS8)。

30

#### 【0062】

その後、プロセッサ100-1内の処理部106においてスレッドBが生成される(ステップS12)。

40

#### 【0063】

スレッドBが生成されると、プロセッサ100-1内のスレッド開始処理部110は、スレッドBを実行待ちキューにつなぐと共に、スレッドBがプロセスDの最初のスレッドであるか否かを判断する(ステップS1)。この場合、スレッドBは最初のスレッドでないので、ステップS2の処理はスキップされる。

#### 【0064】

また、プロセッサ100-1内の並列実行時間算出処理部107において、並列実行時間算出処理が行われる(ステップS13)。このとき、並列カウンタ203の値Cは「0」

50

であるので、並列実行時間テーブル 202 は更新されず、タイムスタンプ 204 の値 TS にシステムタイマ 205 の値 STM が設定される。すなわち、 $TS = 25$  となる (STM は変動しなかったと仮定)。その後、並列カウンタ増加処理部 108 において、並列カウンタ増加処理が行われ、並列カウンタ 203 の値 C が「1」にされる (ステップ S14)。その後、割込み終了処理部 109 による割込み終了処理が行われ、スレッド A はユーザプログラムの処理に復帰する。

#### 【0065】

一方、実行待ちキューにつながれたスレッド B (新たに生成されたスレッド) に、或るプロセッサ (例えば、プロセッサ 100-2) 内のプロセッサ割当処理部 111 によりプロセッサ 100-2 が割り当てられると、プロセッサ 100-2 内の並列実行時間算出処理部 107 が並列実行時間の算出処理を行う (ステップ S13)。この時、スレッド B が属するプロセス D に割り当てられているスレッド共有空間 201 内の並列カウンタ 203 の値 C は「1」であるので、並列実行時間テーブル 202 の第 1 番目のエントリの値 T1 を  $T1 = T1 + (STM - TS) = 15 + (25 - 25) = 15$  とし、タイムスタンプ 204 の値 TS を「25」とする。その後、並列カウンタ増加処理部 108 において、並列カウンタ 203 に対する増加処理が行われ、並列カウンタ 203 の値 C が「2」にされる。

#### 【0066】

その後、割込み終了処理部 109 による割込み終了処理が行われ、スレッド B は、ユーザプログラムの実行を開始する。ここで、スレッド A が割り当てられているプロセッサ 100-1 内の並列実行時間算出処理部 107、並列カウンタ増加処理部 108 よりも、スレッド B が割り当てられるプロセッサ 100-2 内の並列実行時間算出処理部 107、割込み終了処理部 109 が先の処理を行ったとしても最終的な結果は同じである。

#### 【0067】

次に、時刻 P3 ( $STM = 35$ ) において、プロセッサ 100-2 で実行されているスレッド B がスリープシステムコールを発行すると、プロセッサ 100-2 内の割込み開始処理部 102 に制御が移り、並列実行時間算出処理部 103 が並列実行時間の算出処理を行う (ステップ S7)。この時、並列カウンタ 203 の値 C は「2」であるので、並列実行時間テーブル 202 の第 1 番目、第 2 番目の値 T1、T2 が更新される。すなわち、 $STM - TS = 35 - 25 = 10$  が、値 T1、T2 に加算され、 $T1 = T1 + (STM - TS) = 15 + 10 = 25$ 、 $T2 = T2 + (STM - TS) = 0 + 10 = 10$  となる。その後、並列実行時間算出処理部 103 において並列カウンタ 203 の減少処理が行われ、並列カウンタ 203 の値 C が「1」にされる (ステップ S8)。その後、プロセッサ返還処理部 113 がプロセッサの返還処理を行い、更に、ウェイト処理部 112 がスレッド B をスリープキューにつなぎ、停止させる。

#### 【0068】

次に、時刻 P4 ( $STM = 40$ ) において、例えば、プロセッサ 100-2 内のプロセッサ割当処理部 111 が、停止していたスレッド B にプロセッサ 100-2 を割り当てると、プロセッサ 100-2 内の並列実行時間算出処理部 107 が、並列実行時間の算出処理を行う (ステップ S13)。この時、並列カウンタ 203 の値 C は「1」となっているので、並列実行時間算出処理部 107 は、並列実行時間テーブル 202 の第 1 番目のエントリの値 T1 を更新すると共に、タイムスタンプ 204 の値 TS を更新する。すなわち、 $T1 = T1 + (STM - TS) = 25 + (40 - 35) = 30$ 、 $TS = STM = 40$  とする。その後、並列カウンタ増加処理部 108 が、並列カウンタ 203 の値 C を +1 して「2」にする。その後、割込み終了処理部 109 により割込み終了処理が行われ、スレッド B がユーザプログラムの実行を再開する。

#### 【0069】

次に、時刻 P5 ( $STM = 50$ ) において、プロセッサ 100-2 で動作しているスレッド B が、スレッド C を生成するためにスレッド生成システムコールを発行すると、プロセッサ 100-2 内の割込み開始処理部 102 に制御が移る。これにより、並列実行時間算出処理部 103 が並列実行時間の算出処理を行う (ステップ S7)。この時、並列カウン

10

20

30

40

50

タ 2 0 3 の値 C は「2」となっているので、並列実行時間テーブル 2 0 2 の第 1、第 2 番目のエントリの値  $T_1$ 、 $T_2$  に  $(STM - TS)$  を加算し、その後、タイムスタンプ 2 0 4 の値  $TS$  にシステムタイマ 2 0 5 の値  $STM$  を設定する。すなわち、 $T_1 = T_1 + (STM - TS) = 30 + (50 - 40) = 40$ 、 $T_2 = T_2 + (STM - TS) = 10 + (50 - 40) = 20$ 、 $TS = STM = 50$  となる。次に、並列カウンタ減少処理部 1 0 4 が、並列カウンタ 2 0 3 の値  $C$  を 1 減少させ、 $C = 1$  とする。

#### 【0070】

その後、プロセッサ 1 0 0 - 2 内の処理部 1 0 6 でスレッド C が生成される。スレッド C が生成されると、プロセッサ 1 0 0 - 2 内のスレッド開始処理部 1 1 0 は、スレッド C を実行待ちキューにつなぐと共に、スレッド C がプロセス D 内の最初のスレッドであるか否かを判断する（ステップ S 1）。この場合、スレッド C は最初のスレッドでないので、ステップ S 2 の処理はスキップされる。

10

#### 【0071】

また、プロセッサ 1 0 0 - 2 内の並列実行時間算出処理部 1 0 7 において、並列実行時間の算出処理が実行される（ステップ S 1 3）。この時、並列カウンタ 2 0 3 の値  $C$  が「1」であるので、並列実行時間テーブル 2 0 2 の第 1 番目のエントリの値  $T_1$  を更新すると共に、タイムスタンプ 2 0 4 の値  $TS$  を更新する。すなわち、 $T_1 = T_1 + (STM - TS) = 40 + (50 - 50) = 40$ 、 $TS = STM = 50$  となる。次に、並列カウンタ増加処理部 1 0 8 が、並列カウンタ 2 0 3 の値  $C$  を 1 増加させ、 $C = 2$  とする（ステップ S 1 4）。その後、割込み終了処理部 1 0 9 で割込み終了処理が実行され、スレッド B は、ユーザプログラムの処理に復帰する。

20

#### 【0072】

一方、実行待ちキューにつながれている、新たに生成されたスレッド C に或るプロセッサ（例えば、プロセッサ 1 0 0 - 3 とする）内のプロセッサ割当処理部 1 1 1 によりプロセッサ 1 0 0 - 3 が割り当てられると、並列実行時間算出処理部 1 0 7 が、並列実行時間の算出処理を行う（ステップ S 1 3）。この時、並列カウンタ 2 0 3 の値  $C$  は「2」であるので、並列実行時間テーブル 2 0 2 の第 1 番目、第 2 番目のエントリの値  $T_1$ 、 $T_2$  が更新されると共に、タイムスタンプ 2 0 4 の値  $TS$  が更新される。すなわち、 $T_1 = T_1 + (STM - TS) = 40 + (50 - 50) = 40$ 、 $T_2 = T_2 + (STM - TS) = 20 + (50 - 50) = 20$ 、 $TS = STM = 50$  となる。次に、並列カウンタ増加処理部 1 0 8 が、並列カウンタ 2 0 3 の値  $C$  を 1 増加させ、 $C = 3$  とする。

30

#### 【0073】

その後、割込み終了処理部 1 0 9 が割込み終了処理を行い、スレッド C はユーザプログラムの実行を開始する。なお、スレッド B が実行されているプロセッサ 1 0 0 - 2 内の並列実行時間算出処理部 1 0 7、並列カウンタ増加処理部 1 0 8 よりも先に、スレッド C が実行されるプロセッサ 1 0 0 - 3 内の並列実行時間算出処理部 1 0 7、並列カウンタ増加処理部 1 0 8 がステップ S 1 3、S 1 4 の処理を実行したとしても最終的な結果は同じである。

#### 【0074】

次に、時刻 P 6 ( $STM = 60$ ) において、プロセッサ 1 0 0 - 1 で動作しているスレッド A が、終了するために終了システムコールを発行すると、プロセッサ 1 0 0 - 1 内の割込み開始処理部 1 0 2 に制御が移り、並列実行時間算出処理部 1 0 3 が並列実行時間の算出処理を行う（ステップ S 7）。このとき、並列カウンタ 2 0 3 の値  $C$  は「3」であるので、並列実行時間テーブル 2 0 2 の第 1 番目、第 2 番目、第 3 番目のエントリの値  $T_1$ 、 $T_2$ 、 $T_3$  が更新されると共に、タイムスタンプ 2 0 4 の値  $TS$  が更新される。すなわち、 $T_1 = T_1 + (STM - TS) = 40 + (60 - 50) = 50$ 、 $T_2 = T_2 + (STM - TS) = 20 + (60 - 50) = 30$ 、 $T_3 = T_3 + (STM - TS) = 0 + (60 - 50) = 10$ 、 $TS = 60$  となる。その後、並列カウンタ減少処理部 1 0 4 が、並列カウンタ 2 0 3 の値  $C$  を - 1 し、 $C = 2$  とする。

40

#### 【0075】

50

その後、プロセッサ返還処理部 113 が、スレッド A に割り当てていたプロセッサの返還処理を行い、更に、スレッド終了処理部 114 が、スレッド A の終了処理を行う。このとき、スレッド A は、プロセス D 内の最後のスレッドでないので（ステップ S5 が No）、スレッド共有空間 201 の解放処理は行われない。

#### 【0076】

次に、時刻 P7（STM=65）において、プロセッサ 100-3 で動作しているスレッド C が、終了するために終了システムコールを発行すると、プロセッサ 100-3 内の割込み開始処理部 102 に制御が移る。これにより、並列実行時間算出処理部 103 が、並列実行時間の算出処理を行う（ステップ S7）。このとき、並列カウンタ 203 の値 C は、「2」であるので、並列実行時間テーブル 202 の第 1 番目、第 2 番目のエントリの値 T1、T2 が更新されると共に、タイムスタンプ 204 の値 TS が更新される。すなわち、 $T1 = T1 + (STM - TS) = 50 + (65 - 60) = 55$ 、 $T2 = T2 + (STM - TS) = 30 + (65 - 60) = 35$ 、 $TS = STM = 65$  となる。その後、並列カウンタ減少処理部 104 が並列カウンタ 203 の値 C を 1 減少させ、 $C = 1$  とする（ステップ S8）。

10

#### 【0077】

その後、プロセッサ返還処理部 113 が、スレッド C に割り当てていたプロセッサの返還処理を行い、更に、スレッド終了処理部 114 が、スレッド C の終了処理を行う。このとき、スレッド C は、プロセス D 内の最後のスレッドでないので（ステップ S5 が No）、スレッド共有空間 201 の解放処理は行われない。

20

#### 【0078】

次に、時刻 P8（STM=70）において、プロセッサ 100-2 で動作しているスレッド B が、終了するために終了システムコールを発行すると、プロセッサ 100-2 内の割込み開始処理部 102 に制御が移る。これにより、並列実行時間算出処理部 103 が、並列実行時間の算出処理を行う（ステップ S7）。このとき、並列カウンタ 203 の値 C は「1」であるので、並列実行時間テーブル 202 の第 1 番目の値 T1 を更新すると共に、タイムスタンプ 204 の値 TS を更新する。すなわち、 $T1 = T1 + (STM - TS) = 55 + (70 - 65) = 60$ 、 $TS = STM = 70$  となる。その後、並列カウンタ減少処理部 104 が、並列カウンタ 203 の値 C を 1 減算し、 $C = 0$  とする（ステップ S8）。

#### 【0079】

その後、プロセッサ返還処理部 113 がスレッド B に割り当てていたプロセッサの返還処理を行い、更に、スレッド終了処理部 114 がスレッド B の終了処理を行う。この時、スレッド B は、プロセス D 内で最後のスレッドであるので（ステップ S5 が Yes）、スレッド終了処理部 114 は、スレッド共有空間 201 を解放する（ステップ S6）。

30

#### 【0080】

以上の処理によりスレッド A、B、C から構成される並列プログラムの並列実行時間が算出でき、任意の時点で発行する並列実行時間取得システムコールにより、その瞬間の並列実行時間（T1、T2、T3）を容易に取得することが可能となる。例えば、時刻 P8 でスレッド B が並列実行時間取得システムコールを発行した場合、スレッド B は、並列実行時間（ $T1 = 60$ 、 $T2 = 35$ 、 $T3 = 10$ ）を取得することができる。そして、スレッド B は、上記並列実行時間を取得すると、例えば、取得した並列実行時間を表示部（図示せず）に表示したり、並列実行時間に基づいて 2 並列度  $= 35 / 60 = 0.5833$ （約 58.3%）、3 並列度  $= 10 / 60 = 0.16666$ （約 16.7%）を算出し、それらを表示部に表示する。

40

#### 【0081】

なお、上述した実施例においては、主記憶装置 200 上にスレッド共有空間 201 を生成するようにしたが、各プロセッサ 100-1 ~ 100-N によって共有される他の記憶装置上にスレッド共有空間 201 を生成するようにしても構わない。

#### 【0082】

#### 【発明の効果】

50

本発明の第 1 の効果は、マルチプロセッサ計算機上で複数の並列プログラムが動作している場合であっても、各並列プログラムの並列実行時間を取得できるという点である。その理由は、プロセス固有或いはジョブ固有の並列実行時間テーブルを利用して並列実行時間を管理するようにしているからである。

【 0 0 8 3 】

本発明の第 2 の効果は、並列プログラムの実行時の並列度として、2 並列度から N 並列度までの最大 (N - 1) 個の並列度が取得でき、並列度の詳細な検証ができる点である。その理由は、複数のスレッド数 (1, 2, ..., N) 毎に、そのスレッド数以上のスレッドに同時にプロセッサが割り当てられた時間を登録する領域を有する並列実行時間テーブルを用いて、並列実行時間を管理しているからである。

10

【図面の簡単な説明】

【図 1】本発明の実施例のブロック図である。

【図 2】図 1 の処理例を示す流れ図である。

【図 3】具体例を挙げて実施例の動作を説明するための図である。

【符号の説明】

1 0 0 - 1 ~ 1 0 0 - N ... プロセッサ

1 0 1 ... ユーザプログラム実行処理部

1 0 2 ... 割込み開始処理部

1 0 3 ... 並列実行時間算出処理部

1 0 4 ... 並列カウンタ減少処理部

1 0 5 ... 並列実行時間取得処理部

1 0 6 ... 処理部

1 0 7 ... 並列実行時間算出処理部

1 0 8 ... 並列カウンタ増加処理部

1 0 9 ... 割込み終了処理部

1 1 0 ... スレッド開始処理部

1 1 1 ... プロセッサ割当処理部

1 1 2 ... ウェイト処理部

1 1 3 ... プロセッサ返還処理部

1 1 4 ... スレッド終了処理部

2 0 0 ... 主記憶装置

2 0 1 ... スレッド共有空間

2 0 2 ... 並列実行時間テーブル

2 0 3 ... 並列カウンタ

2 0 4 ... タイムスタンプ

2 0 5 ... システムタイマ

K ... 記録媒体

20

30

